ELECTRONIC DOCUMENT CLASSIFICATION USING SUPPORT VECTOR MACHINE-AN APPLICATION FOR E-LEARNING

Sandeep Dixit
NPTI Badarpur New Delhi India

L.K.Maheshwari
BITS Pilani Rajasthan India

Keywords: Support Vector Machines (SVM), Term Frequency (TF), Term Frequency – Inverse Document Frequency (TF – IDF).Internet services, Dial-up networking.

Abstract: Appplication of the machine learning techniques to embed adaptivity in the E-Learning frameworks is receiving considerable attention. Text Classification, or the task of automatically assigning semantic categories to natural language text, has therefore become one of the key methods for organizing digital content.Reports on SVM have mainly focussed on the theory or conceptual application of the SVM with little, if any, concern to applicability to E-Learning domain. In this paper we present a holistic approach towards building a text classifier suited for E-Learning domains. We present theory, application and use of a specific SVM software(SVM *Torch*) for classification of electronic documents giving every detail of how the downloaded SVM software can be used to apply the SVM concept and categorise text document in the context of E-Learning domains. Experimental results obtained by applying SVM to text document are presented. *Pre-processing* of the document is also presented. The experiment was conducted using SVM *Torch* software with ten documents for training and five documents for testing. SVMs performed the best when used with binary representation. We are confident that the extent of details provided could well serve as a useful component in E-Learning frameworks and even provide curriculum component for UG level students.

1 INTRODUCTION

Text Classification, or the task of automatically assigning semantic categories to natural language text, has become one of the key methods for organizing online information. Since hand-coding classification rules is costly or even impractical, most modern approaches employ machine learning techniques to automatically learn text classifiers from examples. Other text classification techniques reported are the word based soft clustering Algorithm,[King-Ip Lin, Ravi Kumar, Dept. of science university of Memphis, USA]where a comparative study of the quality of clusters by K-Means, WBSC, Buckshot and Fractionation has bee shown. However the tuning of parameters of the algorithms in these studies is left out. Another text classification technique reported (.H.Li, et.al.in *The Computer Journal, Vol.41, No.8, 1998l*) uses Naïve Bayesian, Neural Networks, Decision Trees, and Subspace Classifier. However use of SVM has been left out in such applications specifically, even though the high dimensionality is efficiently handled with SVM. Support Vector Machines (SVM) theory is finding increasing application in the classification of the documents as it combines high performance and efficiency with theoretical understanding and improved robustness. In particular, it is highly effective without greedy heuristic components and computationally efficient in training and classification. With SVMs the generalization error gets optimised and the mean and standard deviation of the error are improved. In this paper we have used the '*SVMTorch by Collobert and Bengio*' for experimentation as it could work even with very high dimensional data.

Dixit S. and K.Maheshwari L. (2005).
ELECTRONIC DOCUMENT CLASSIFICATION USING SUPPORT VECTOR MACHINE-AN APPLICATION FOR E-LEARNING.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 191-198

The paper is organised as follows: Section I provides an overview of SVM, its working and how it lends itself for the type of categorisation work necessary in e-learning paradigm, Section II provides a brief description of the SVM Torch software. Section III describes the approach used and the pre-processing of the document. Section IV gives the complete implementation of the Text categorisation using the SVM software and gives experimental results and the analysis. The method for representing the documents and the use of SVM *Torch* in classification of text documents has been dealt therein. The code for the document pre-processing, commands to be executed for training SVM and classification of documents and sample screen shots have also been included.

2 SECTION I

2.1 Support Vector Machines

The foundations of Support Vector Machines (SVM) have been developed by Vapnik (Vapnik,V.,1995) and are gaining popularity due to many attractive features, and promising empirical performance. The formulation embodies the Structural Risk Minimisation (SRM) principle, which is superior (Gunn, S.R., et al., 1997), to traditional Empirical Risk Minimisation (ERM) principle, employed by conventional neural networks. SRM minimises an upper bound on the expected risk, as opposed to ERM that minimises the error on the training data. This is the difference that equips SVM with a greater ability to generalise, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to the domain of regression problems (Vapnik,V., et al., 1997). The term SVM is typically used to describe classification with support vector methods and support vector regression is used to describe regression with support vector methods. Kernel Selection is an important issue and the obvious question that arises is that with so many different mappings to choose from, which is the best for a particular problem? This is not a new question, but with the inclusion of many mappings within one framework it is easier to make a comparison. The upper bound on the VC dimension is a potential avenue to provide a means of comparing the kernels. However, it requires the estimation of the radius of the hypersphere enclosing the data in the non-linear feature space.The problem of empirical data modelling is germane to many engineering applications. In empirical data modelling a process of induction is used to build up a model of the system, from which it is hoped to deduce responses of the system that are yet to be observed. Ultimately the quantity and quality of the observations govern the performance of this empirical model. By its observational nature data obtained is finite and sampled. This sampling is non-uniform and due to the high dimensional nature of the problem the data will form only a sparse distribution in the input space. Consequently the problem is nearly always ill posed (Poggio,T. et al., 1985) in the sense of Hadamard (Hadamard,J., 1923). Traditional neural network approaches have suffered difficulties with generalisation, producing models that can over fit the data. This is a consequence of the optimisation algorithms used for parameter selection and the statistical measures used to select the 'best' model. As a final caution, even if a strong theoretical method for selecting a kernel is developed, unless this can be validated using independent test sets on a large number of problems, methods such as bootstrapping and cross-validation will remain the preferred method for kernel selection.

2.2 SVM and Document Classification

The emergence of Student Centred Learning Models in E-Learning has led to lecture presentation as per the learner's choice and state of knowledge. In this context text categorization has become an area of intense research. Several methods have been proposed for such purpose(Y.H.Li, et.al.in *The Computer Journal, Vol.41, No.8, 1998*) but we chose Support Vector Machines as they can be used efficiently for document classification by using a feature representation and a kernel method that best represents the features. The process of representing the document by a feature representation is termed as *pre-processing*. The features need to be mapped to numerals as SVM software work with numerals. Support Vector Machine is essentially calculated as weighted sum of kernel function outputs. The kernel itself could be polynomial, RBF Gaussian or any other function satisfying Mercer's conditions. We used the polynomial as the kernel function. Conceptually the Lagrangian formed with the objective function (constructed from the SVM output expression) is minimised and the classification margin maximised for the training set.

2.3 Feature representation

A feature is a word in the document classification. A feature vector consists of the various words from a dictionary formed by analysing the documents. There are various alternatives and enhancements in

constructing the feature vectors. Some of them are (Drucker,H. et al., 1999):

- Term Frequency (TF)
- Term Frequency Inverse Document Frequency (TF-IDF)

The i^{th} component of the feature vector is the number of times that the word appears in the document. Sometimes, the feature vector is normalized to unit length Binary Representation.TF-IDF uses the above TF multiplied by the IDF (inverse document frequency). The Document Frequency (DF) is the number of times that a word occurs in all the documents or some (based on specifications). The inverse document frequency is defined as

$IDF(w_i) = \log(|D|/DF(w_i)),$
Mod D is number of documents.

Binary Representation

Binary Representation represents whether a particular word occurs in a particular document presentation In addition to the any of the above, stop words may or may not be used. Words like “of”, “and”, “the”, etc., are used to form the stop list. Words on the stop list are not used in forming a feature vector. The argument against the use of stop words is that it is not obvious which words, beyond the trivial, should be on the stop list

Polynomial Kernel

A polynomial mapping is a popular method for non-linear modeling,

$$K(x, x') = \langle x, x' \rangle^d.$$

The kernel used is usually the second equation, a slightly modified form with addition of 1 to the right hand side of the above expression as it avoids problems with the Hessian becoming zero.

$$K(x, x') = \left(\langle x, x' \rangle + 1\right)^d.$$

3 SECTION II

The SVMTorch Software

SVMTorch is the software tool used for using SVMs for classification. *SVMTorch* (Source: www.support-vector.com) is a new implementation of Vapnik's Support Vector Machine that works both for classification and regression problems, and that has been specifically tailored for large-scale problems (such as more than 20000 examples, even for input dimensions higher than 100).

Commands

SVMTorch software enables user to train the SVM and use it for classification with the help of two commands. They are *SVMTorch* and *SVMTest*. The command SVMTorch is used for training the SVM using the training data set and the command SVMTest is used for classifying the test documents. The syntax of both these commands are given below

The general syntax:

- SVM*Torch SVMTorch [options] example_file model_file*
- SVM*Test SVMTest [options] model_file test_file*

where "example_file" is the file obtained from the training set file after preprocessing, "test_file" is the file obtained from the testing set file after preprocessing and "model_file" is the SVM-model created by SVMTorch.

Options

SVMTorch provides the above commands with multiple options. All the options are described when SVMTorch or SVMTest is launched without any argument. By default, SVMTorch is a classification machine. If the regression machine is needed, the option –rm is to be used. The current error displayed by SVMTorch is only an indicator. It can oscillate.

SVMTorch

The major options of the command *SVMTorch* are:
- *multi* <int> for multiclass problems. The data must be in multiclass format.
-*mlc* <*int*> for multiclass problems. The last class to learn can be specified here.
-*c* <*float*> <*float*>is the trade-off between training error and margin. Defult=100.
- *unshrink* if the optimality of removed variables at the end of learning phase is to be checked. It will possibly restart the optimization. The SVMTorch will be slower with this option.
-*d* <*int*> where <*int*> is the parameter *d* in the polynomial kernel.
- *std* <*float*> where <*float*> is the parameter *std* in the Gaussian kernel.
- *s* <*float*> where <*float*> is the parameter *s* in the sigmodial or in the polynomial kernel.

- *r* <*float*> where <*float*> is the parameter *r* in the sigmodial or in the polynomial kernel.
- *u* <*string*> where <*string*> is a parameter for the user defined kernel.
- *t* <*int*> <*int*> defines kernel:
  0 is the *linear* kernel
  1 is the *polynomial* kernel (a,b)
  2 is the *gaussian* kernel
  3 is the *sigmoidal* kernel

SVMTest

The options of the command *SVMTest* are:
- *multi* for multiclass problems. The data should also be in the same format.
-*norm* when in multiclass mode, this option normalizes (with the inverse of the margin norm) the output of each model before deciding in which class the example should be. To avoid time conconsuming the margin is now approximated by the sum of the α_i where α_i is the coefficient related to i^{th} support vector. (In the separable case it is the same as the margin)
- *sparse* for the sparse mode. This is used when the data is in the sparse format.
- *bin* if the data is in binary, set this flag.
- *load* <int> loads only <int> examples in the test set.
- *oa* <file> prints the output of the model in <file>, in ASCII.
- *ob* <file> prints the output of the model <file>, in binary..

File format

The files that are given to the above commands can in be various formats. The default format of SVMTorch and SVMTest is the ASCII format. Therefore, -*bin* is to be given in the command line while using binary files. No tests are made with binary files, and it could have an error such as "Check your data" or "Segmentation fault". Also, -*sparse* is to be given if the data is in sparse format.
There are two main input formats for "input_file" and "test_file" in SVMTorch. They are:

- ASCII format
- Binary format

The general format of a file in ASCII format is given below.

```
 <Number n of training/testing
samples>  <Dimension d of each
sample+1>
<a11>  <a12>  <a13>   ....  <a1d> <a1_out>
 .
 .
 .
<an1>  <an2>  <an3>   ....  <and> <an_out>
```

where <a_{ij}> is an ASCII floating point number corresponding to the j^{th} value of the i^{th} example and <a_{i_out}> is the i^{th} desired output (in classification, it should be +1/-1). The binary format is the same, without carriage returns.There is another special input format for SVM*Test*, when there is no desired output. (-*no* option is to be used).

4 SECTION III

Programming and categorization approach

This section explains the coding and use of SVM Torch and mentions the steps to be followed for the classification of documents. Broadly the two aprts are The problem posed is shown in the Table 1.and gives the approach used by us

Esentially, preprocessing involves conversion of the text document into a file with format supported by SVM for classification. The format used is *ASCII mode- Standard*. The training documents as well as the test documents must be preprocessed. The preprocessing can be done using any one of the alternatives present (Ex: Term frequency, Term frequency-Inverse Document frequency, Binary representation etc.) along with defining some rules for building the library. We have developed programs for preprocessing using the techniques Term frequency (TF), Term frequency-Inverse Document frequency (TF-IDF), Binary representation along with
The following points must be strictly followed while executing any of the above programs:

- The files given as input to any of the above mentioned programs are to be in the same directory as that of the program being executed.
- The number of input files should be less than 20.
- The value given as input for class should be either 1 or -1 as we are dealing with only binary classification.

In case of pre-processing of training files, no. of files corresponds to no. of training samples (documents) and class to which category the document belongs while in case of processing for test files, no. of files corresponds to the documents to be classified and class corresponds to the expected category to which it is to be classified.

In case of pre-processing with the help of rules, an additional input of the value of rule is to be provided. This value represents the minimum

Table 1: Gives an overview of the approach used in using SVM *Torch.*

INPUT : A set of text documents based on a kew descriptor of lecture content
OUTPUT: Classify the favourable documents against a mix of other text documents.
STEPS
1.Preprocessing
1.1Documents for both training and testing are converted into a file with ASCII format (which is supported by SVMTorch)
The following steps are involved in converting the documents into a file with ASCII format. This stage also involves the removal of articles and irrelevant terms
1.1.1 A dictionary is built based on the training files.
1.1.2 All the documents are represented in the form of feature vector.
1.1.3.These feature vectors are written into a file.
2.Classification
2.1.Takes the preprocessed output from previous stage .and uses them for training and testing the SVM.

number of times the word should occur (when counted it's occurrence in all the documents).

Classification Stage

The files generated from the preprocessing stage are used for training and testing the SVM by SVM*Torch* software with the help of the two commands SVMTorch and SVMTest mentioned previously..
The code has been developed in C and tested in UNIX and LINUX platforms. For executing the commands given in the previous section for training and testing the SVM, SVM*Torch* software should be downloaded and all the files mentioned above should be present in the directory SVM*Torch*
Once the preprocessing is done, the classification of the test documents could be done by training the Support Vector Machine using the file created out of the training documents. For training the Support Vector Machine, any of the available software could be used (with proper changes in the file created out of the training documents). We have used SVMTorch software for this purpose. It is free software downloaded from the net. It provides two commands for training the SVM and for classifying the documents using SVM. They are:

- SVM*Torch*
- SVM*Test*

These two commands are provided with several options

Training

SVMTorch is the command used for training the SVM. Its syntax is given below:

```
SVMTorch [option] example_file
model_file
```

where "example_file" is the file generated from training examples and "model_file" is the output model generated by the command.

Testing

SVMTest is the command used for classifying the documents using SVM. Its syntax is given below

```
SVMTest [options] model_file test_file
```

where "model_file" is the output model generated by *SVMTorch* and "test_file" is the file generated from the test documents

5 SECTION IV

Experimental Results

For experimentation purpose we have included text from diverse categories of documents so as to

Table 2: Results of categorization.

Contents of Test file	Expected Classification	Actual Classification
Bitmap Indexing	1	1
September 11 attacks	-1	-1
Climatic Conditions	-1	-1
Bitmap Indexing	1	1
Bitmap Indexing	1	-1

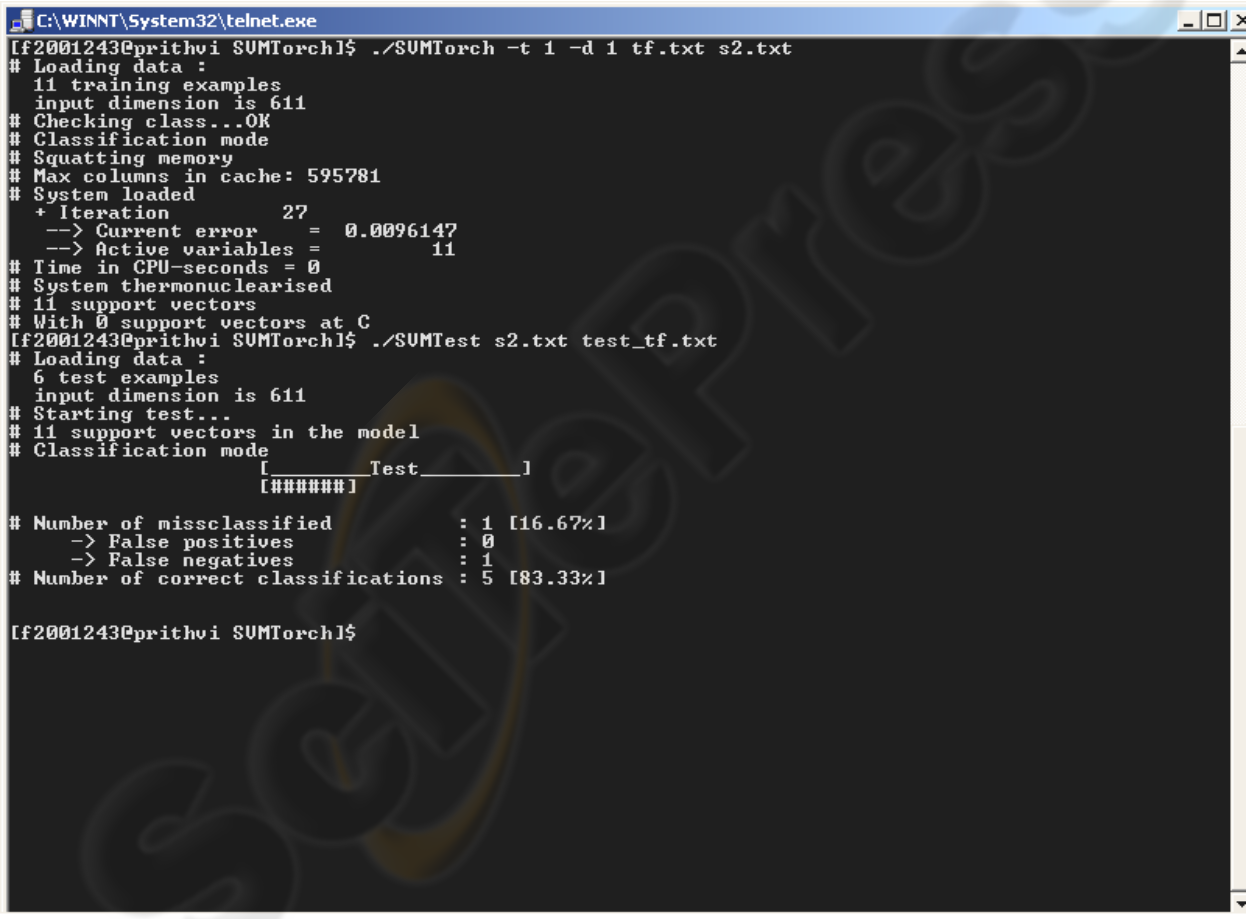

Figure 1: Sample Screenshot. It shows one of the ouput samples. Number of Support Vectors is *11*, Input dimension being *611*(small).

Table 3: SVM application for E-Learning framework-Learner Centered Model

Student Needs	SVM role for classification of the Digital content for E-Learning Paradigm
Review selected text only (before exams) Provides the keyword	Classified and Categorize Slides, Text using SVM
All about 'Router'	Categorize and produce select set of documents from lecture on 'Computer networks'using SVM
What are good candidate documents for reviewing on 'Concept X'	Present a list of documents classified based on the given concept category.
Rank Lecture documents based on relevance to coming exam	Prioritise the documents so as to optimise on the time to be given for each document before a presentation or talking the exam
Q & A Session, e.g.*'What is an SVM ?'*	Lookup the lecture sets for the string with the given 'Question' keyword and classify that document for further presentation.

subject the SVM to worst testing environment. The categories cover 'bitmap indexing', 'climate conditions' etc.

The screenshot in fig.1.above shows the runtime environment while the table 2 indicates the success results obtained on testing the SVM trained on ten documents using first order polynomial as the kernel function. Of the ten input documents, five are about Bitmap indexing and the remaining files are about irrelevant matter.

E-Learning Application Scenario

Here we briefly present the use of SVM in the context of larger perspective of learner centered E-Learning frameworks wherein adaptivity and student/learner context is to be perceived prior to the delivery of content. The table 3 above provides a brief glimpse of one such possible framework. The digital content takes the form of Text(slides, books), Graphics, videos and the student requirement is to extract only the needed material.Such techniques are making the E-learning systems increasingly dynamic with features exhibiting the human tutor characteristics. The various documents and data relevant to a course of study are digitally preserved and on request from the user, these data are presented in the available form with minimal processing. The learner seeks to acquire tailor made information as per his choice of keyword. It is here that the appropriate document classification must take place so that the classified document forms the input to the overall personalized E-Learning system.

Analysis

The paucity of space did not permit us to give complete details. However it is important to note that text classification is a high dimensional problem and speeds as well as accuracy are major concerns. Due to the capability of the SVM to produce acceptable generalization error even with small size of training data, they are expected to be better than the ML techniques like NN, BN etc. at least in the matter of text classification. Text classification poses a challenge due to its unique characteristics like–the vector size being proportional to the vocabulary, and the rare words having same weight as normal words, etc. In the case of subjects in engineering stream these characteristics are further accentuated due to multi dimensional nature of 'term meaning' depending on context. Therefore it is necessary to employ an efficient technique such as SVM for categorization of text. The time for the classification in the tested documents is shown as zero due to small size of the document and hence the dimension

6 CONCLUSION

In this paper we have presented the detailed use of SVM and SVM *Torch* software for categorization of document in the larger context of E-Learning frameworks. The results were slightly optimistic due to smaller number and size of test documents. The number of support vectors was 11 and the dimension was 611.The reports of other techniques used for classification like Naïve Bayesian, Neural Networks, Decision Trees also provide encouraging results. However the SVM approach to classification is expected to be superior from the point of view of size of training data particularly for text classification tasks where high dimensionality and speed are major concerns. Specifically, SVMs could yield acceptable generalization errors even with small size training data as compared to the

techniques like NN and BN. The use of Reduced Set SVM could further enhance the speed of the classification. Our next work covers the effect of *natural languages and context* on the SVM vector size and dimensionality issues.

REFERENCES

Vapnik,V.,1995 The Nature of Statistical Learning Theory,

Gunn, S.R., Brown, M., and Bossley, K.M.,1997, Network performance assessment for neurofuzzy data modeling, Intelligent Data Analysis, volume 1208 of Lecture Notes in Computer Science, pages 313–323.

Vapnik,V., Golowich,S., Smola,A., 1997 Support vector method for function approximation, regression estimation, and signal processing, In M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pages 281–287, Cambridge, MA, MIT Press.

Poggio,T., Torre,V.,Koch,C.,1985 Computational vision and regularization theory, Nature, 314–319.

Hadamard,J.,1923, Lectures on the Cauchy Problem in Linear Partial Differential Equations, Yale University Press.

Vapnik,V., 1998, Statistical Learning Theory, Springer, N.Y.

Minoux,M.,1986, Mathematical Programming: Theory and Algorithms, John Wiley and Sons.

Drucker,H., Donghui Wu, Vapnik,V.N.,1999, Support Vector Machines for Spam Categorization, Proceedings of the IEEE Transaction on Neural Networks, vol. 10, no. 5, September.

Aronszajn,N.,1950 Theory of reproducing kernels, Trans. Amer. Math. Soc., 686:337–404.

Girosi,F.,1997, An equivalence between sparse approximation and Support Vector Machines, A.I. Memo 1606, MIT Artificial Intelligence Laboratory.

Heckman, N.,1997, The theory and application of penalized least squares methods or reproducing kernel hilbert spaces made easy.

Wahba,G.,1990 Spline Models for Observational Data. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia.

www.support-vector.com.